

Appl. No. 09/784,392
Amdt. dated June 2, 2004
Reply to Office of April 5, 2004

CLAIMS

The listing of claims below replace all prior versions, and listings, of claims:

- 1 1. (Previously Presented) A method of performing a transaction in a database system, comprising:
 - 3 receiving a transaction to be performed, wherein the transaction is
 - 4 processed by a plurality of access modules; and
 - 5 performing a flush of a transaction log from volatile storage to non-
 - 6 volatile storage by each access module before an end transaction procedure.

- 1 2. (Previously Presented) The method of claim 1, further comprising issuing a request to flush the transaction log with a message sent to each access module for performing a last step of the transaction, the last step performed prior to the end transaction procedure.

- 1 3. (Previously Presented) The method of claim 2, further comprising performing the flush of the transaction log in a data access step prior to the end transaction procedure to avoid performance of a transaction log flush in the end transaction procedure.

- 1 4. (Previously Presented) The method of claim 2, further comprising determining that the last step is being performed by all of the plurality of access modules involved in the transaction.

- 1 5. (Original) The method of claim 1, further comprising determining if the transaction log has been flushed before performing the end transaction procedure.

- 1 6. (Original) The method of claim 5, further comprising avoiding performance of a transaction log flush in the end transaction procedure if the transaction log has been flushed.

Appl. No. 09/784,392
Amdt. dated June 2, 2004
Reply to Office of April 5, 2004

- 1 7. (Original) The method of claim 1, further comprising:
2 identifying the transaction as an implicit transaction.

- 1 8. (Original) The method of claim 1, further comprising:
2 performing the end transaction procedure, which follows execution of the
3 transaction.

- 1 9. (Original) The method of claim 8, performing the end transaction
2 procedure comprising:
3 skipping broadcast of a directive indicating commencement of the end
4 transaction procedure to the plurality of access modules.

- 1 10. (Original) A method of performing an end transaction procedure in a
2 database system, comprising:
3 a first access module in the database system writing an end transaction
4 indication to a first transaction log portion, the first access module being part of a
5 cluster of access modules; and
6 the first access module sending an end transaction directive to a
7 fallback module associated with the first access module, the fallback module being
8 part of the cluster.

- 1 11. (Original) The method of claim 10, wherein the first access module sends
2 the end transaction directive to the fallback module but not to other access modules in the
3 cluster.

- 1 12. (Original) The method of claim 10, wherein sending the end transaction
2 directive comprises sending an end transaction-part one directive.

- 1 13. (Original) The method of claim 12, further comprising the first access
2 module broadcasting an end transaction-part two directive to all access modules in the
3 cluster.

Appl. No. 09/784,392
Amdt. dated June 2, 2004
Reply to Office of April 5, 2004

1 14. (Original) The method of claim 10, further comprising the fallback
2 module writing an end transaction indication to a second transaction log portion.

1 15. (Previously Presented) The method of claim 10, further comprising the
2 first access module flushing the first transaction log portion from volatile storage to non-
3 volatile storage.

1 16. (Original) The method of claim 10, further comprising the first access
2 module flushing the first transaction log portions but the other access modules in the
3 cluster not flushing their respective transaction log portions.

1 17. (Previously Presented) A database system comprising:
2 a plurality of storage media, the storage media comprising persistent
3 storage;
4 volatile storage; and
5 a plurality of access modules, wherein each access module is coupled to
6 one of the plurality of storage media; and
7 each of the access modules being adapted to flush a transaction log from
8 the volatile storage to the persistent storage before performing an end transaction
9 procedure.

1 18. (Original) The database system of claim 17, further comprising a
2 controller adapted to determine if each access module has flushed the transaction log
3 maintained by the access module.

1 19. (Original) The database system of claim 18, wherein the controller is
2 adapted to skip sending a directive to perform a transaction log flush if the controller
3 determines that each access module has flushed the transaction log before the end
4 transaction procedure.

Appl. No. 09/784,392
Amdt. dated June 2, 2004
Reply to Office of April 5, 2004

1 20. (Previously Presented) The database system of claim 17, further
2 comprising a controller adapted to provide a flush directive with a message to each of the
3 access modules to perform a last step of the transaction before the end transaction
4 procedure.

1 21. (Previously Presented) An article comprising a medium storing
2 instructions for enabling a processor-based system to:
3 receive a transaction to be performed, wherein the transaction is processed
4 by a plurality of access modules;
5 determine that a last step of the transaction involves the plurality of access
6 modules, wherein the last step is performed before an end transaction procedure; and
7 flush a transaction log from volatile storage to a non-volatile storage while
8 the last step is performed by the plurality of access modules.

1 22. (Previously Presented) The article of claim 21, further storing instructions
2 for enabling the processor-based system to:
3 perform the end transaction procedure, wherein the end transaction
4 procedure follows execution of the last step of the transaction.

1 23. (Previously Presented) The article of claim 22, further storing instructions
2 for enabling a processor-based system to:
3 avoid broadcast of a directive indicating commencement of the end
4 transaction procedure to the plurality of access modules.

1 24. (Previously Presented) A method of performing a transaction in a database
2 system, comprising:
3 receiving a transaction to be performed on plural access modules in the
4 database system;
5 maintaining a log in volatile storage to track operations performed in the
6 transaction; and

Appl. No. 09/784,392
Amdt. dated June 2, 2004
Reply to Office of April 5, 2004

7 writing the log to persistent storage before start of an end transaction
8 procedure.

1 25. (Original) The method of claim 24, wherein writing the log to persistent
2 storage comprises flushing the log.

1 26. (Original) The method of claim 24, wherein maintaining the log comprises
2 maintaining a transaction log.

1 27. (Original) The method of claim 24, further comprising performing the end
2 transaction procedure, the end transaction procedure comprising writing an end
3 transaction indication into the log.

1 28. (Previously Presented) A database system comprising:
2 storage media comprising persistent storage;
3 volatile storage;
4 access modules coupled to the storage media; and
5 a parsing engine coupled to the access modules, the parsing engine
6 adapted to perform one of:

7 (a) providing a directive with a message to perform a last step
8 of a transaction and communicating the directive to the access modules, each access
9 module responsive to the directive to perform a transaction log flush from the volatile
10 storage to the persistent storage before performance of an end transaction procedure; and

11 (b) determining if each of the access modules has performed a
12 transaction log flush before start of the end transaction procedure;

13 the parsing engine adapted to avoid sending a broadcast directive to the
14 access modules to cause performance of a transaction log flush during the end transaction
15 procedure.

Appl. No. 09/784,392
Amdt. dated June 2, 2004
Reply to Office of April 5, 2004

1 29. (Previously Presented) The method of claim 1, wherein the transaction
2 comprises plural steps, the method further comprising:

3 performing the plural steps prior to performing the end transaction
4 procedure, and

5 wherein performing the flush of the transaction log comprises performing
6 the flush of the transaction log in one of the plural steps.

1 30. (Previously Presented) The method of claim 29, wherein performing the
2 plural steps comprises performing, in each of the plural steps, access of relational table
3 data stored in the database system.

1 31. (Previously Presented) The method of claim 30, wherein performing the
2 flush of the transaction log in one of the plural steps comprises performing the flush of
3 the transaction log in a last one of the plural steps.

1 32. (Previously Presented) The method of claim 31, further comprising each
2 access module adding a first entry to the transaction log to redo the transaction by the
3 access module in case of system failure.

1 33. (Previously Presented) The method of claim 4, wherein performing the
2 flush of the transaction is prior to the end transaction procedure if the last step is
3 performed by all of the plurality of access modules, the method further comprising:

4 performing the flush of the transaction log in the end transaction
5 procedure if the last step is not performed by all of the plurality of access modules.

1 34. (Previously Presented) The database system of claim 17, wherein the
2 access modules to perform a transaction comprising plural steps, one or more of the
3 access modules adapted to perform the plural steps prior to the end transaction procedure,
4 and the access modules adapted to perform the flush of the transaction log in one of the
5 plural steps.

Appl. No. 09/784,392
Amdt. dated June 2, 2004
Reply to Office of April 5, 2004

1 35. (Previously Presented) The database system of claim 34, wherein the one
2 of the plural steps comprises a last one of the steps.

1 36. (Previously Presented) The database system of claim 35, wherein the
2 transaction log comprises a first entry associated with each access module to enable a
3 redo of the transaction in case of system failure.

1 37. (Previously Presented) The database system of claim 36, wherein the
2 transaction log further comprises a second entry associated with each access module to
3 enable an undo of the transaction.

1 38. (Previously Presented) The database system of claim 34, further
2 comprising a controller to determine whether a last one of the steps involves all the
3 access modules, and in response to determining that the last one of the steps involves all
4 the access modules, the controller to send a directive to all the access modules to perform
5 the flush of the transaction log in the last one of the steps.

1 39. (Previously Presented) The database system of claim 38, in response to
2 determining that the last step does not involve all access modules, the controller to send a
3 directive to perform the flush of the transaction log in the end transaction procedure.

1 40. (Previously Presented) The article of claim 21, wherein the transaction
2 comprises plural steps, the article further storing instructions for enabling a processor-
3 based system to:

4 perform the plural steps prior to performing the end transaction procedure,
5 and

6 wherein performing the flush of the transaction log comprises performing
7 the flush of the transaction log in one of the plural steps.

Appl. No. 09/784,392
Amdt. dated June 2, 2004
Reply to Office of April 5, 2004

1 41. (Previously Presented) The article of claim 40, wherein performing the
2 plural steps comprises performing, in each of the plural steps, access of relational table
3 data stored in the database system.

1 42. (Previously Presented) The article of claim 41, wherein performing the
2 flush of the transaction log in one of the plural steps comprises performing the flush of
3 the transaction log in a last one of the plural steps.

1 43. (Previously Presented) The article of claim 42, further storing instructions
2 for enabling a processor-based system to cause each access module to add a first entry to
3 the transaction log to redo the transaction by the access module in case of system failure.